Appendice al Cap. 5

Decomposizione LU di una matrice quadrata

Una qualunque matrice quadrata $M = \{m_{ij}\}$ di ordine N, reale, invertibile, i cui minori principali siano tutti non nulli, si può sempre decomporre come

$$M = LU\,,$$

dove $L = \{\alpha_{ij}\}$ e $U = \{\beta_{ij}\}$ sono matrici triangolari i cui elementi nulli si trovano, rispettivamente, in basso a sinistra (L : *lower triangular*) e in alto a destra (U : *upper triangular*) :

$$L = \begin{pmatrix} \alpha_{11} & 0 & 0 & \cdots & 0 \\ \alpha_{21} & \alpha_{22} & 0 & \cdots & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \cdots & 0 \\ \vdots & & & \cdots & \vdots \\ \alpha_{N1} & \alpha_{N2} & \alpha_{N3} & \cdots & \alpha_{NN} \end{pmatrix}, \; U = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \cdots & \beta_{1N} \\ 0 & \beta_{22} & \beta_{23} & \cdots & \beta_{2N} \\ 0 & 0 & \beta_{33} & \cdots & \beta_{3N} \\ \vdots & & & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{NN} \end{pmatrix}. \tag{5.1}$$

Gli elementi α_{ij}, β_{ij} e m_{ij} sono tra loro legati dal sistema di equazioni

$$m_{ij} = \sum_{k=1}^{N} \alpha_{ik}\beta_{kj}\,. \tag{5.2}$$

Il numero di equazioni che occorre soddisfare è pari al numero di elementi di L e U diversi da zero. Sovrapponendo le due matrici si vede facilmente che si tratta di $N^2 + N$ dal momento che N^2 sono quelli della matrice L sommati a quelli della matrice U esclusa la diagonale ed N sono proprio questi ultimi.

Siccome le relazioni a disposizione sono solo N^2 abbiamo la possibilità di scegliere arbitrariamente i valori di N elementi. Una scelta semplice consiste nell'imporre che tutti i termini della diagonale di L siano pari a 1 :

$$\alpha_{ii} = 1\,.$$

Spezziamo ora la somma al secondo membro del sistema (5.2)in tre termini :

$$m_{ij} = \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj} + \alpha_{ii}\beta_{ij} + \sum_{k=i+1}^{N} \alpha_{ik}\beta_{kj} \,. \tag{5.3}$$

I termini della somma piú a destra sono tutti nulli perché per essi $k > i$ e, in questo caso, $\alpha_{ik} = 0$. I termini con $k > i$ sono infatti quelli per cui l'indice di colonna è maggiore dell'indice di riga, quindi sono quelli che nella matrice L si trovano nel triangolo a destra in alto.

Portando a secondo membro m_{ij} e a primo membro β_{ij}, e ricordando che $\alpha_{ii} = 1$, l'equazione (5.3) diventa

$$\beta_{ij} = m_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj} \,. \tag{5.4}$$

Quest'equazione vale per ogni $i \leqslant j$. Per $i > j$ gli elementi β_{ij} sono nulli. Usiamo ancora l'espressione (5.3), interrompendo la prima sommatoria per $k < j$:

$$m_{ij} = \sum_{k=1}^{j-1} \alpha_{ik}\beta_{kj} + \alpha_{ij}\beta_{jj} + \sum_{k=j+1}^{N} \alpha_{ik}\beta_{kj} \,.$$

In questo caso il terzo addendo della somma è nullo perché sono nulli gli elementi β_{kj} con $k > j$, cioè quelli il cui indice di riga è maggiore dell'indice di colonna. In altre parole sono nulli i termini che si trovano nel triangolo in basso a sinistra. Riorganizzando i termini si ottiene dunque

$$\alpha_{ij} = \frac{1}{\beta_{jj}} \left(m_{ij} - \sum_{k=1}^{j-1} \alpha_{ik}\beta_{kj} \right) \,. \tag{5.5}$$

È facile rendersi conto che l'espressione sopra riportata vale per $j > i$, essendo tutti gli altri termini nulli. Per realizzare questo algoritmo basta iniziare a calcolare gli elementi della prima colonna di L e U. Infatti $\alpha_{11} = 1$ e, una volta calcolato $\beta_{11} = m_{11}$, si può calcolare α_{21} che richiede solo i primi termini per poter essere determinato. In effetti, scrivendo l'equazione (5.5) per $i = 2$, $j = 1$ si ottiene

$$\alpha_{21} = \frac{m_{21}}{\beta_{11}} \,,$$

e cosí via. Una possibile realizzazione dell'algoritmo in questione dunque è quella riportata nel Listato 5.1. In questo listato la sequenza delle operazioni da eseguire sugli elementi di matrice è stata opportunamente riscritta in modo da lasciare invariato il risultato, ottimizzando il codice.

```
/* assegna i valori della diagonale di L */
for (i = 0; i < N; i++) {
  alpha[i][i] = 1.;
}

/* calcola gli elementi fuori della diagonale */
for (j = 0; j < N; j++) {
  for (i = 0; i <= j; i++) {
    beta[i][j] = m[i][j];
    for (k = 0; k <= i - 1; k++) {
      beta[i][j] -= alpha[i][k]*beta[k][j];
    }
  }
  for (i = j + 1; i < N; i++) {
    alpha[i][j] = m[i][j];
    for (k = 0; k <= j - 1; k++) {
      alpha[i][j] -= alpha[i][k] * beta[k][j];
    }
    alpha[i][j] /= beta[j][j];
  }
}
```

Listato 5.1 Algoritmo di Crout per la decomposizione LU di una matrice quadrata.

La decomposizione LU è uno strumento utilissimo nell'algebra lineare. Innanzi tutto cosituisce un metodo alternativo a quello di Gauss per la soluzione dei sistemi di equazioni lineari. Il sistema

$$Mx = y\,,$$

infatti, si può scrivere come $LUx = y$. Se si definisce $Ux = z$, si può risolvere prima il sistema $Lz = y$, da cui si ricava z e quindi il sistema $Ux = z$ da cui si ricava x. La soluzione dei due sistemi le cui matrici sono triangolari è banale, perché basta iniziare dalla riga con un solo elemento e *salire* o *scendere* con l'indice di riga per trovare tutti i valori delle variabili (vedi anche il Paragrafo 5.4 del libro).

Una volta decomposta una matrice $M = LU$, calcolarne il determinante è un'operazione semplicissima. Il determinante di una matrice gode della proprietà per cui

$$\det(M) = \det(L)\det(U)\,,$$

ma $\det(L) = 1$ e il determinante della matrice U è dato dalla produttoria degli elementi della sua diagonale.

Infine osserviamo che l'inversa della matrice M si trova come prodotto delle inverse delle matrici L ed U. Il calcolo dell'inversa è semplificato dal fatto che l'inversa di una matrice triangolare è ancora una matrice triangolare, perciò si ha che, se chiamiamo $\bar{\alpha}_{ij}$

gli elementi della matrice L^{-1}, $\bar{\alpha}_{ij}\ \forall j > i$. Scriviamo dunque il sistema di equazioni cui devono soddisfare gli elementi di L^{-1}, tenendo conto di quest'osservazione.

$$\sum_{k=1}^{i-1} \alpha_{ik}\bar{\alpha}_{kj} + \alpha_{ii}\bar{\alpha}_{ij} = \delta_{ij}\,.$$

Nel caso in cui $i = j$ gli addendi della sommatoria a sinistra sono tutti nulli, perché sono nulli i fattori α_{ik} per $i > k$. Da quest'osservazione si ricava che

$$\bar{\alpha}_{ii} = \frac{1}{\alpha_{ii}}\,.$$

Potremmo porre $\alpha_{ii} = 1$, ma per il momento manteniamo l'espressione piú generale. Per $i < j$ invece si deve avere che

$$\bar{\alpha}_{ij} = -\frac{1}{\alpha_{ii}} \sum_{k=1}^{i-1} \alpha_{ik}\bar{\alpha}_{kj}$$

e i diversi termini si ottengono applicando iterativamente questa formula. Tutti gli elementi $\bar{\alpha}_{kj}$ necessari a calcolare $\bar{\alpha}_{ij}$, infatti, sono già noti dal momento che $k < i$.

Il codice che corrisponde al calcolo degli elementi di matrice inversa di L è riportato nel Listato 5.2, in cui la variabile `invL` rappresenta l'inversa di L.

```
1  for (i = 0; i < N; i++) {
2    invL[i][i] = 1./alpha[i][i];
3    for (j = 0; j < i; j++) {
4      invL[i][j] = 0;
5      for (k = 0; k <= i - 1; k++) {
6        invL[i][j] -= alpha[i][k] * invL[k][j];
7      }
8      invL[i][j] /= alpha[i][i];
9    }
10 }
```

Listato 5.2 Calcolo dell'inversa di L.

Per calcolare l'inversa di U basta osservare che la sua trasposta è una matrice triangolare sinistra, come L. Basta dunque applicare lo stesso algoritmo scambiando `alpha` con `beta` e gli indici `i` e `k` nel prodotto righe per colonne, come nel Listato 5.3. Questo spiega perché conviene lasciare nell'espressione dell'inversa di L gli elementi α_{ii} : nel caso della matrice U gli elementi diagonali non sono uguali a 1 e non potremmo usare lo stesso codice. Nel listato abbiamo chiamato la variabile che contiene la matrice U^{-1} `invU`.

```
1  for (i = 0; i < N; i++) {
2    invU[i][i] = 1./beta[i][i];
```

```
3     for (j = 0; j < i; j++) {
4       invU[i][j] = 0;
5       for (k = 0; k <= i - 1; k++) {
6         invU[i][j] -= beta[k][i] * invU[k][j];
7       }
8       invU[i][j] /= beta[i][i];
9     }
10  }
```

Listato 5.3 Calcolo dell'inversa di U.

Se conoscete le funzioni è facile scriverne una che sia utile per invertire entrambe le matrici, cosí come è utile, per verificare che non ci siano errori nel codice, scrivere una funzione che calcoli il prodotto righe per colonne tra due matrici e usarla per controllare che il prodotto di una matrice per la sua inversa sia uguale alla matrice unità.